Assignment 9 – Queue Using List

CS 244

Due: April 10, 2014, 11:59 PM

Late penalties will be as described in the syllabus.

Overview

First implement a template singly linked class.
Then implement a template Queue class by using the template singly linked list class.

General Objectives:

Explore: How to implement a template class Queue using a singly linked list.
In the previous assignments you learned about dynamic arrays, std::vector and how to use it to create a template stack. Here you will gain practice creating a singly linked list, and how to create a template Queue class.

First Step

In Linux in your …/Documents/Programs folder
Create a folder named ***A09***
with 2 subfolders: /***PartA_LinkedList***
/***PartB_Queue***

This is necessary as you will compress the folder and its contents for submission.

Second Step Check D2L

There will be starter code for this assignment. Likely posted near where this document was found. It may make things easier, and some of the files are explicitly required.

For Part A the files must be in A09/PartA_LinkedList and MUST be named:

SingList.h **SLTester.cpp** **makefile**

For Part B the files must be in A09/PartB_Queue and MUST be named:

Queue.h **QueueTester.cpp** **makefile**

with the corresponding classes properly defined and declared in each. The makefile files will be provided for you and ***should not*** require any changes.

All files (excluding makefiles) should have comments with the file name, your name, the last modified date, and a description of the file contents. All functions should be appropriately commented.

This assignment will be partially graded by automated tester programs.
Test your code thoroughly.

-=-=-

Assignment 9 – Queue Using List CS 244

Part A – Implement a Singly Linked List

The starter code should include 3 files in the A09 subfolder named: PartA_LinkedList:

SingLink.h SLTester.cpp makefile

This Part A must be completed before starting part B, as part B needs the SingList.h file to be functioning correctly.

Most of the changes you are required to make occur in the SingLink.h file.
The makefile file should require no changes.
The SLTester.cpp should only require additional testing to be added by you.
It contains some simple tests for you to expand upon.

In the SingList.h file you will find a template NodeType class. No changes should be needed to it. You will also find a template SingList class. The declaration portion of this class should require no changes. Below the declaration of the SingList class you will find the area you must make changes to. Your goal is to implement each function of the SingList class correctly and use the code in the main() function found in SLTester.cpp to test your implementation.

Also on D2L, in Homework->Examples, you may find another file and folder named something like:

SinglyLinkedList/SinglyLinkedList.cpp

This might offer some tips on how to finish this assignment.

The general outline of what each function should do follows. Details of implementation are up to you to decide, discover, and test what is and is not necessary to create a functioning singly linked list class.

SingList constructor: Initialize mp_HeadPtr and mp_TailPtr to null.

SingList destructor: Needs to delete the list. Possibly using the class helper function deleteList(). mp_headPtr and mp_tailPtr should also be set to null.

insertFront: Should insert the node pointed to by newNode onto the front of the list. Do NOT make a copy of the node. Remember to adjust all the pointers correctly.

removeFront: Removes the first node from the list (and frees its allocated memory)

insertBack: Same as insertFront but adds the node to the end of the list

removeBack: Removes the last node from the list (and frees its allocated memory). Remember to adjust all pointers correctly.

front: returns a copy of the data stored in the first node in the list

printList: prints the contents of the list. It may or may not include an endl

deleteList: removes all nodes from the list. Adjusts all pointers correctly. Frees all memory correctly.

empty: returns true if the list is empty, else returns false

size: returns the number of elements in the list

Check the declaration section of SingList.h to verify that is all of the required functions to implement.

Assignment 9 – Queue Using List CS 244

Part B – Implement a Queue Using the a Singly Linked List from Part A

The starter code should include 3 files in the A09 subfolder named: PartA_LinkedList:

Queue.h QueueTester.cpp makefile

Part A must be completed before starting part B, as this part B needs the SingList.h file to be functioning correctly. Note the included path to it in Queue.h.

Most of the changes you are required to make occur in the Queue.h file.
The makefile file should require no changes.
The QueueTester.cpp should only require additional testing to be added by you.
It contains some simple tests for you to expand upon.

In the Queue.h file you will find a template Queue class. The declaration portion of this class should require no changes. Below the declaration of the Queue class you will find the area you must make changes to. Your goal is to implement each function of the Queue class correctly and use the code in the main() function found in QueueTester.cpp to test your implementation.

The general outline of what each function should do follows.
Note this is just an outline. Details of implementation are up to you to decide, discover, and test what is and is not necessary to create a functioning queue class.

Queue constructor: not much to do here, but think on it

Queue destructor: not even listed in the declaration section, likely not needed

enqueue: Insert the newNode onto the back of the queue. Suggest you call a related function using member variable m_list

dequeue: Remove the first node in the queue

front: return a copy of the data stored in the first node in the queue

printQueue: print all the elements in the queue

empty: return true if the queue is empty, else return false

size: return the number of elements in the queue

This will be subject to automated grading (at least partially), so function and filenames matter.

-=-=-=

Assignment 9 – Queue Using List CS 244

Grading

100 points possible

At least 80 of these points will be determined by an "automatic grading" program.

The other 20 will be determined from code examination and/or further automated grading.

The majority of the points will be for correctly implementing the template SingList and Queue classes. Not as many points will be given for your main() testing functions. Those are for you to estimate what your grade will be before turning it in. It is likely a separate testing file created by the instructor will be used to test your class(es). The better you write your own testing program the more likely you will score well on the instructor's.

Bonus Points

There may be in-class work that can be submitted for bonus points to be applied to this assignment. However, total score will not be increased beyond the 100 points possible.

Turn-In Directions

Correctly submitting your work is worth 0 points,
but if not done correctly will likely result in nothing to grade.

Preparation

In Ubuntu Linux browse to your A09 folder
Make sure your source code files are in the folder
Right click on the A09 folder,
Set the file name to be A09_*yourlastname*.tar.gz
where *yourlastname* is your last name
Example: if your last name is Gollygee
then the filename would be A09_Gollygee.tar.gz
Select compress
This should create the file named A09_ *yourlastname*.tar.gz

Submit

Submit the A09_ *yourlastname*.tar.gz file
to the correct course drop box in D2L